REMARKS

Reconsideration of this application as amended is respectfully requested. This Amendment is submitted in response to the Office Action mailed April 7, 2011, which was made final. Claims 1, 2, 4-10, 12-17, and 19-21 are rejected. In this Amendment, claims 1, 4-6, 9, 15, and 19 have been amended. Claims 7 and 8 have been canceled without prejudice. No new claims have been added. The specification has been amended to correct typographical errors. It is respectfully submitted that the amendment does not add new matter. Applicants reserve all rights with respect to the applicability of the Doctrine of Equivalents.

Interview Summary

The applicants thank the Examiner for the courtesy of the telephone interview with an attorney for applicants on April 5, 2011. An Interview Summary was mailed by the Examiner on April 7, 2011. The Applicants would like to summarize the matters discussed during the interview. During the interview, the Examiner agreed that the application would be allowable if the independent claims were amended to include translating from a 32-bit program to a 64-bit platform. No agreement was reached at the time of the interview.

Claim Rejections Under 35 U.S.C. § 101

The Examiner rejects claims 15-18 under 35 U.S.C. §101 as being directed to non-statutory subject matter. Applicants respectfully submit that claim 15 has been amended to include, for example, a memory and a processor that performs various functions. Applicants respectfully submit that amended claim 15 falls under one of the four statutory categories of invention under 35 U.S.C. § 101. Consequently, claims 16-18, which depend from claim 15,

also fall under one of the four statutory categories of invention under 35 U.S.C. § 101. Applicants respectfully request withdrawal of the rejection of claims 15-18 under 35 U.S.C. §101.

Claim Rejections Under 35 U.S.C. § 103

The Examiner rejects claims 1-2, 4-5, and 19-21 under 35 U.S.C. § 103(a) as being unpatentable over U.S. Publication No. 2005/0120194 by Kissell in view of U.S. Publication No. 2003/0126313 by Kerly. Applicants reserve the right to swear behind Kissell at a later date. Applicants respectfully request withdrawal of these rejections because the cited reference fails to teach or suggest all of the features of the claims.

Kissell describes a fork instruction for execution on a multithreaded microprocessor while occupying a single instruction issue slot. (Kissell, Abstract). Kissell further describes multithreaded processors including multiple sets of resources, or contexts, for storing unique states of each thread, thereby facilitating the ability to quickly switch between threads to fetch and issue instructions. (Kissell, paragraph 7). If the resources are not available to execute a new thread context, an exception is raised (Kissell, Figure 5).

Kerly describes a system to reduce thrashing in a multi-threaded environment by intercepting thread creation. (Kerly, Abstract). The thread is created for modification of a thread stack, and an initial stack pointer is modified. (Kerly, paragraph 0041). The thread stack pointer is modified by an offset and the first function in the stack is executed. (Kerly, paragraph 0042; Figure 7).

Amended claim 1 recites:

> A computer implemented method comprising:

beginning initialization of a first thread from a second platform, wherein the first thread is executable on a first platform and the second platform, wherein the first thread is 32-bit platform-independent code;

suspending the initialization of the first thread at a position within the second platform, wherein the beginning initialization of the first thread is suspended in response to a detection of an operating system request to create the first thread;

creating a second thread based on the position in the second platform;

completing the initialization of the first thread continuing from the position in the second platform; and

translating the 32-bit platform-independent code for execution on the second platform, wherein the second platform is a 64-bit platform.

(Emphasis Added)

Applicants respectfully submit that a combination of Kissell and Kerly fails to teach or suggest "translating the 32-bit platform-independent code for execution on the second platform, wherein the second platform is a 64-bit platform," as recited in claim 1.

As discussed above, Kissell describes a fork instruction for execution on a multithreaded microprocessor while occupying a single instruction issue slot. (Kissell, Abstract). Kissell further describes multithreaded processors including multiple sets of resources, or contexts, for storing unique states of each thread, thereby facilitating the ability to quickly switch between threads to fetch and issue instructions. (Kissell, paragraph 7). Although Kissell describes switching between threads to fetch and issue instructions and interrupts for service requests, Kissell is silent as to performing any thread translations, let alone "translating the 32-bit platform-independent code for execution on the second platform, wherein the second platform is a 64-bit platform," as claimed.

Furthermore, Kerly describes a system to reduce thrashing in a multi-threaded environment by intercepting thread creation. (Kerly, Abstract). The thread is created for modification of a thread stack, an initial stack pointer is modified by an offset, and the first function in the stack is executed. (Kerly, paragraphs 0041-0042; Figure 7). Kerly, however, also

fails to teach or suggest "translating the 32-bit platform-independent code for execution on the second platform, wherein the second platform is a 64-bit platform," and fails to remedy the shortcomings of Kissell set forth above.

Therefore, applicants respectfully submit that a combination of Kissell and Kerly fails to teach or suggest "translating the 32-bit platform-independent code for execution on the second platform, wherein the second platform is a 64-bit platform," and fails to render claim 1, and the claims that depend therefrom, obvious.

Amended claim 19 recites:

> A computer system for managing thread resource comprising:
> a random accessed memory;
> a first processor configured to execute multithreaded programs stored in the random accessed memory;
> a multithreaded program to be executed on a second processor; and
> a program to transparently initialize, create, and translate a thread included in the multithreaded program in an environment supported by the first processor to be executed on the second processor, wherein the initialization of the thread is suspended in response to a detection of an operating system request to create the thread, and wherein the thread is 32-bit platform independent code and is translated for execution on a 64-bit platform.

(Emphasis Added)

Applicants respectfully submit that a combination of Kissell and Kerly fails to teach or suggest "a program to transparently initialize, create, and translate a thread included in the multithreaded program in an environment supported by the first processor to be executed on the second processor ... wherein the thread is 32-bit platform independent code and is translated for execution on a 64-bit platform," as recited in claim 19.

As discussed above, Kissell describes a fork instruction for execution on a multithreaded microprocessor while occupying a single instruction issue slot. (Kissell, Abstract). Kerly describes a system to reduce thrashing in a multi-threaded environment by intercepting thread

creation. (Kerly, Abstract). However, neither Kissell nor Kerly, alone or in combination, teaches or suggests performing the claimed thread translations. Therefore, a combination of Kissell and Kerly also fails to teach or suggest "a program to transparently initialize, create, and translate a thread included in the multithreaded program in an environment supported by the first processor to be executed on the second processor ... wherein the thread is 32-bit platform independent code and is translated for execution on a 64-bit platform," as recited in claim 19.

Therefore, applicants respectfully submit that a combination of Kissell and Kerly does not render claim 19, and the claims that depend therefrom, obvious.

Applicants respectfully request withdrawal of the rejection of claims 1-2, 4-5, and 19-21 under 35 U.S.C. § 103(a) as being unpatentable over Kissell in view Kerly.

The Examiner rejects claims 6-18 under 35 U.S.C. § 103(a) as being unpatentable over Kissell in view of Kerly, and further in view of an article entitled "64-bit versus 32-bit Virtual Machines for Java" by Venstermans, et al (hereinafter "Venstermans"). Applicants reserve the right to swear behind Venstermans at a later date.

As discussed above, with respect to claim 1, a combination of Kissell and Kerly fails to teach or suggest each and every limitation as claimed. Venstermans teaches the behavior characteristics of 32-bit and 64-bit Java virtual machines (Venstermans, Abstract), but fails to remedy the shortcomings of Kissell and Kerly set forth above. Given that claims 6 depends from claim 1, and include additional features and limitations, applicants respectfully submit that the rejection of claims 6 under 35 U.S.C. § 103(a) has been overcome for at least the same reasons as described above and respectfully requests the withdrawal of the rejection of the claims.

Amended claim 9 recites:

A computer implemented method comprising:

beginning initialization of a foreign thread from a host platform, wherein the foreign thread is to be executed on a foreign platform and the foreign thread is 32-bit platform independent code;

suspending the initialization of the foreign thread at a position within the host platform, wherein the beginning initialization of the foreign thread is suspended in response to a detection of an operating system request to create the foreign thread;

recording the position of the suspension;

creating a host thread from a host platform based on the recorded position;

completing the initialization of the foreign thread continuing from the recorded position in the host platform; and

translating the 32-bit platform-independent code for execution on the host platform, wherein the host platform is a 64-bit platform.

(Emphasis Added)

Applicants respectfully submit that a combination of Kissell, Kerly, and Venstermans fails to teach or suggest "translating the 32-bit platform-independent code for execution on the host platform, wherein the host platform is a 64-bit platform," as recited in claim 9.

As discussed above, Kissell describes a fork instruction for execution on a multithreaded microprocessor while occupying a single instruction issue slot (Kissell, Abstract), Kerly describes a system to reduce thrashing in a multi-threaded environment by intercepting thread creation (Kerly, Abstract), and Venstermans teaches the behavior characteristics of 32-bit and 64-bit Java virtual machines (Venstermans, Abstract). However, none of Kissell, Kerly, and Venstermans, alone or in combination, teaches or suggests performing any thread translations, let alone "translating the 32-bit platform-independent code for execution on the host platform, wherein the host platform is a 64-bit platform," as claimed.

Therefore, applicants respectfully submit that a combination of Kissell, Kerly, and Venstermans does not render claim 9, and the claims that depend therefrom, obvious.

Amended claim 15 recites:

A computer system for managing thread resources comprising:
a memory;

a processor coupled with the memory;

a first multithreaded programming environment, executed by the processor;

a second multithreaded programming environment, executed by the processor;

a multithreaded program, executed by the processor, including a first thread and a second thread, wherein the first thread is 32-bit platform independent code;

a host platform, wherein the host platform is a 64-bit platform; and

a dynamic binary translator to translate the first thread for the first multithreaded programming environment to be executed in the second multithreaded programming environment, wherein the binary translator further includes an operating system wrapper configured to suspend operating system requests from the first thread when requests to create a new thread are detected, and wherein the translation includes translation of the 32-bit platform independent code for execution on the 64-bit platform.

(Emphasis Added)

Applicants respectfully submit that a combination of Kissell, Kerly, and Venstermans fails to teach or suggest "a dynamic binary translator to translate the first thread for the first multithreaded programming environment to be executed in the second multithreaded programming environment ... wherein the translation includes translation of the 32-bit platform independent code for execution on the 64-bit platform," as recited in claim 15.

As discussed above, Kissell describes a fork instruction for execution on a multithreaded microprocessor while occupying a single instruction issue slot (Kissell, Abstract), Kerly describes a system to reduce thrashing in a multi-threaded environment by intercepting thread creation (Kerly, Abstract), and Venstermans teaches the behavior characteristics of 32-bit and 64-bit Java virtual machines (Venstermans, Abstract). However, none of Kissell, Kerly, and Venstermans, alone or in combination, teaches or suggests performing any thread translations, let alone "a dynamic binary translator to translate the first thread for the first multithreaded programming environment to be executed in the second multithreaded programming

environment ... wherein the translation includes translation of the 32-bit platform independent code for execution on the 64-bit platform," as claimed.

Therefore, applicants respectfully submit that a combination of Kissell, Kerly, and Venstermans does not render claim 15, and the claims that depend therefrom, obvious.

Applicants respectfully request withdrawal of the rejection of claims 6-18 under 35 U.S.C. § 103(a) as being unpatentable over Kissell in view of Kerly, and further in view of Venstermans.

The Examiner rejects claims 15-19 under 35 U.S.C. § 103(a) as being unpatentable over U.S. Publication No. 2006/0184920 by Wang, et al (hereinafter "Wang") in view of Kerly. Applicants respectfully reserve the right to swear behind Wang at a later date.

Wang describes a method and apparatus to support the execution of a managed application that is linked to a native library or application (Wang, Abstract). Wang describes a translator that may include an interface and a dynamic binary translator. The interface defines an application program interface (API) that enables communication between a virtual machine and the dynamic binary translator. Although Wang describes a dynamic binary translator, Wang does not teach or suggest "a dynamic binary translator to translate the first thread for the first multithreaded programming environment to be executed in the second multithreaded programming environment, wherein the binary translator further includes an operating system wrapper configured to suspend operating system requests from the first thread when requests to create a new thread are detected, and wherein the translation includes translation of the 32-bit platform independent code for execution on the 64-bit platform" (emphasis added), as claimed.

Furthermore, as discussed above, Kerly fails to teach or suggest the performing any thread translations, and thus fails to teach or suggest "a dynamic binary translator to translate the

first thread for the first multithreaded programming environment to be executed in the second multithreaded programming environment ... wherein the translation includes translation of the 32-bit platform independent code for execution on the 64-bit platform," as recited in amended claim 15.

For at least the reasons set forth above, applicants respectfully request withdrawal of the rejection of claims 15-19 under 35 U.S.C. § 103(a) as being unpatentable over U.S. Publication No. 2006/0184920 by Wang in view of Kerly.

Conclusion

Applicant respectfully submits that in view of the amendments and discussion set forth herein, the applicable rejections have been overcome. Accordingly, the present and amended claims should be found to be in condition for allowance.

If a telephone interview would expedite the prosecution of this application, the Examiner is invited to contact the undersigned at (408) 720-8300.

If there are any additional charges/credits, please charge/credit our deposit account no. 02-2666.

Respectfully submitted,
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

Dated: August 8, 2011

/William L. Jaffe/
William L. Jaffe
Reg. No. 64,977

Customer No. 45209
1279 Oakmead Parkway
Sunnyvale, CA 94085
(408) 720-8300